UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/919,753	08/01/2001	Joseph T. Apuzzo	POU900182US1	6371

7590 08/16/2004

Kevin P. Radigan, Esq.
HESLIN & ROTHENBERG, P.C.
5 Columbia Circle
Albany, NY 12203

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 08/16/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/919,753	APUZZO ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 August 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *01 August 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *8/1/2001*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed August 1, 2001.

Claims 1-54 have been submitted for examination.

Specification

2. The disclosure is objected to because of the following informalities: the element referred to as 'mapped expected results therefore' (e.g. pg.3, bottom, para 0009, pg. 4) includes the term 'therefore' and this appears to be a misprint or a grammatical misuse. Such term should be corrected to be 'therefor' in order to give the element some sense.

Appropriate correction is required.

Claim Objections

3. Claims 1, 15, 20, 34, 35, 40, 41, and 51 are objected to because of the following informalities: "... expected results therefore."(line 7, 13, 10, 8, 14, 12, 11, and 12, respectively). The term 'therefore' appears to be a misprint and should be 'therefor' in order for the limitation to make sense. Appropriate correction is required.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Okayasu, USPN: 5,659,554 (hereinafter Okayasu), in view of Schmidt, USPubN: 2002/0026630 (

hereinafter Schmidt) and JAD, "Test Case Identification JAD Session", 7/2001(hereinafter JAD); further in view of Brouwer et al., USPN: 6,279,124 (hereinafter Brouwer), and Ottensooser, USPN: 5,905,856 (hereinafter Ottensooser).

As per claim 1, Okayasu discloses a method of testing a software component, comprising:

creating an abstraction matrix that describes the software component, comprising state and event information (e.g. col. 3, line 23-40 – Note: state transition diagram being parsed implicitly teaches a well-known concept such as a 2-D representation state-transition table – see col. 10, lines 41-45, i.e. a matrix); parsing the abstraction matrix to generate test cases (e.g. Fig. 3, 4; Fig. 11);

But Okayasu does not explicitly disclose separating test cases based on layers of the software component; nor does Okayasu disclose associating data structures with the test cases, the data structures such that these allow the layer test cases to be uncorrelated. Okayasu mentions about a complex large-scale system with hierarchy of states and parallel transitions (col. 3, lines 42-67) and identifying of a process represented by a state (Fig. 11); hence has suggested large system coordination of test cases with software process/state association in a large and complex software system. The organization of large software system into hierarchy of modules with functional requirements allotted to each sub-modules composing the hierarchy and representing test case with test script or structure program was a known concept at the time the invention was made. JAD, for example, identifies functional, performance and informational requirements by which to allocate test cases, and identifying of modules and coverage of test cases therefor and test case matrix (e.g. pg. 2-3); while Schmidt, in a system using test cases in a

integration system with state prediction similar to Okayasu, discloses generating test cases in light of state diagrams according to levels of the infrastructure (see para 0504-0508, pg. 16). All of which suggest the organizing of a target software system into required subparts and levels to base test implementation thereupon. Hence, It would have been obvious for one of ordinary skill in the art at the time the invention was made to organize complex system software as mentioned by Okayasu so that test cases are layered according to the hierarchy of software modules as known in the art and as suggested by JAD and Schmidt because this would facilitate the modular control of test case administration for the same manner as software system are modularized into hierarchy or layer of sub-modules composing a upper module. According to this rationale, for each test case, a script would be imparted to one module of the large software system so that running the test script per layer of the large system would appear to be uncorrelated to another test case pertaining to another module. Hence, providing data structures in form of scripts representing the test cases would enable the layer-structured test case system to perform in more controllable and modularized manner for the benefits known in the art of modularizing a large and complex software system as suggested by Okayasu.

Okayasu does not teach employing the component in executable form to generate test case execution threads for each layer and executing in parallel some of the execution threads for at least one of the layer, thereby testing the software component. Testing a large system in a complex software system as approached by Okayasu has been enhanced by the layer-based test cases as set forth above. The effect of parallel transitions and hierarchized state behaviors in Okayasu's (enhanced by Schmidt and JAD) complex software system can be furthered by executing the test scripts or programs in a synchronized manner and consecutively so for time-

efficiency, e.g. threads execution. Brouwer, in a method to generate test cases like Okayasu and Schmidt/JAD, discloses test case automation and synchronized control of test scripts execution with support of multi-layered APIs and threads mutual exclusion (e.g. col. 13, line 30 to col. 17, line 8). It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a executable form of the component (Note: the use of a preliminary or pre-built form of executable of a target software component/module is inherent in any test of software) and executes it in form of parallel and synchronized threads as taught by Brouwer because this would automate the test system in controlled and time-efficient manner so that error detecting, control code and corrective adjustment for improvement can be imparted in a well-behaved fashion (see Brouwer: Background).

Nor does Okayasu disclose generating mapped expected results upon parsing the abstraction matrix and mapped expected results to the step of executing test case threads. The process of setting up expected results during analysis of functional requirements and mapping of the expected results to the test results was part of requirement analysis and product validation, and was a well-known concept at the time the invention was made. Ottensooser, in a method to generate test plan with test scripts analogous to paradigm provided by the combination Okayasu and Schmidt/JAD, discloses associating expected output to each test scripts and comparing of expected outputs after running of test plan (col. 2, line 14 to col. 3, line 63). Hence, in view of the well-known concepts of establishing expected results and use them to map against the test outputs, it would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance the test script/case generation method of Okayasu and Schmidt/JAD (enhanced with Brouwer) so that the test execution results would be compared with expected

results gathered at test scripts generation time as taught by Ottensooser, because this would enable accommodation to intangible variable conditions incurred during various stages of test; and by coupling a set of expected outputs per test script being generated, this would provide validation assessment of results and possibly negate the effects of the intangible conditions per test execution (see Ottensooser: Background).

As per claim 2, Okayusa discloses a functional specification (e.g. step S2 –Fig. 11, 16).

As per claim 3, Okayusa discloses parsing, separating, employing and executing being automated (e.g. Fig. 1, 3, 6, 11, 16 and see combination in claim 1)

As per claims 4 and 5, see rationale using Brouwer parallel threads execution when executing test scripts in parallel in view of the combination of Okayasu and Schmidt/JAD. The motivation for executing parallel threads per layer or submodule or per multiple submodules would have been obvious for the same reasons or purposes.

As per claim 6, Schmidt discloses more than one test cases or scripts per system level or scenario (e.g. para 0789, pg. 23); hence the motivation to combine Okayasu with Schmidt to enable multiple test case per layer would have been obvious by virtue of the benefits as set forth in claim 1.

As per claim 7, Okayusa only disclose a abstraction matrix (e.g. col. 10, lines 41-45); but according to the teachings by JAD using a matrix to map requirements and allocating test cases to software modules or functional requirements representing aspects of the business functions (pg. 2-3); and by Schmidt/JAD (re claim 1) for addressing organizing test case into layers or subparts or modules, which was a known concept at the time of the invention, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement such

abstraction matrix by Okayusa with state and event transition so that it reflects test case being imparted per layer of the software system or per component of the hierarchy of modules as suggested by JAD and Schmidt/JAD, because of the same rationale as set forth in claim 1.

As per claim 8, official notice is taken that using a file to represent a level of mathematical abstraction, or algorithmic computation or pseudo-code was a known concept at the time of the invention (e.g. Mathworks file, m-file by MathCad); hence in view of the teachings and rationale in claim 7, wherein a large system require organizing requirement in controllable components and test script per scenario of validating those identified components of the functional requirements, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the abstraction matrix as taught by Okayusa, so that each abstraction matrix would be stored as a file because this would facilitate management control and persistency storage with inherent file system control benefits so well known in a complex software development and management system.

As per claim 9, Schmidt discloses input parameters and expected results (e.g. para 0319, pg. 11; para 0620, pg. 19) and Ottensooser discloses input parameter in association with a particular transition or functions the system is designed to have as well as expected output (e.g. col. 2, lines 29-38). In view of the large system to design as suggested by Okayusa, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement test cases based on functional requirements such as attributes or parameter per functions along with expected outputs would be used as inputs into the test case generation process as suggested by Schmidt and Ottensooser because of a well-known concept as to why test cases is purported, i.e. to support validation and fulfill as much as possible, all the fawcetts

and attribute of the submodules or parts of the software system; in view of their expected behaviors.

As per claims 10 and 11, Okayusa discloses matrix with state and event information for each state of the component and event that leads a state to a next state (e.g. Fig. 1, 4, 17-19).

As per claim 12, official notice is taken that designing complex software system such as operating system with multi-layered kernel was a known concept in the art. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to apply the software component test case as taught by Okayusa (combined with Schmidt/Ottensooser/ Brouwer and JAD) so that it is for supporting and implementing as complex a system as an OS taught by the above notice, because only complex systems require establishing of complex and elaborate test scenarios to support in-depth validation of all the aspects of the complex hierarchy of functions making up the system, and according to a well-known concept, an operating system is one of those complex multi-layered system.

As per claim 13, Okayasu discloses analysis of state transition and determining possible transitions associating a process (Fig. 11, 16) but does not teach a minimum number of states required to model the software component. But in view of the teachings by JAD and Schmidt to implement the software in terms of functional requirements, the process of identifying the minimum or eventually a maximum of features required for the software is implicitly disclosed. Hence, the motivation to establish this minimum number of functions or states analysis as suggested by Okayasu, in light of the scheme of preparing the requirement analysis via the implicit teachings by JAD or Schmidt, would have been obvious because according to well-known concept of keeping a design to an optimal cost level, the providing of a bare minimal

number of required features as implicitly derived from JAD or Schmidt would help cut down the expenditure allocated for building the software system.

As per claim 14, Okayasu provides an engine to associate relationship between transitions (e.g. unit 4, 5 - Fig. 4) while Schmidt for example discloses scripts for supporting test cases, and gathering of models in preparing for team testing (pg. 16-17, para 0487-0548) to provide association between sub-modules or level to be tested; and running test in parallel (para 0822, pg. 24); hence has suggested providing relationship in creating validation test and apparent non-correlation between test script execution. Since Okayasu suggests a complex system with parallel process state and Brouwer teaches threads synchronization as set forth in claim 1, the motivation as to provide the scripts and parallel and undependent execution of test execution as mentioned by Schmidt or Brouwer to Okayasu's method would be for the same reasons as mentioned in claim 1 using Brouwer to provide thread-save execution and time efficiency.

As per claim 15, Okayasu discloses a method of generating test cases for a software component, comprising a functional specification of the software component (e.g. step S2 –Fig. 11, 16); creating an abstraction matrix using said specification (col. 10, lines 41-45), the matrix comprising state and event information (col. 3, line 23-40); and parsing the matrix to generate test cases (e.g. Fig. 3, 4; Fig. 11).

But Okayasu does not disclose generating mapped expected results while parsing the matrix; nor does Okayasu disclose separating test cases based on layers of the software component; nor does Okayasu disclose associating data structures with the test cases, the data structures such that these allow the layer test cases to be uncorrelated. But all these limitations have been addressed in claim 1; and will be referred thereto for the corresponding rejections.

As per claims 16-19, refer to rejections as set forth in claims 3, 8, 11, and 14, respectively.

As per claim 20, this is an system claim including limitations corresponding to those of method claim 1, hence will be rejected using the corresponding rejection as set forth therein.

As per claims 21-33, refer to rejections as set forth in claims 2-14, respectively.

As per claim 34, Okayasu discloses a system for testing a software component employing an abstraction matrix describing the component, and comprising state and event information (e.g. col. 10, lines 41-45; col. 3, line 23-40), the system comprising: an abstraction engine for parsing the abstraction matrix and generating test cases (Fig. 3, 4; Fig. 11).

But Okayasu does not disclose generating mapped expected results while parsing the matrix; nor does Okayasu disclose separating test cases based on layers of the software component; nor does Okayasu disclose associating data structures with the test cases, the data structures such that these allow the layer test cases to be uncorrelated. Nor does Okayasu teach employing the component in executable form to generate test case execution threads for each layer and executing in parallel some of the execution threads for at least one of the layer, thereby testing the software component.

But all these limitations have been addressed in claim 1; and will be referred thereto for the corresponding rejections.

As per claims 35-39, these claims correspond to the method claims 15-19, hence are rejected using the corresponding rejection as set forth therein, respectively.

As per claim 40, Okayasu discloses a system for generating test cases for a software component, comprising storage medium for storing an abstraction matrix (e.g. col. 10, lines 41-

45), a functional specification of the software component (e.g. step S2 –Fig. 11, 16); the matrix being created using said specification (e.g. step S2 –Fig. 11, 16), the matrix comprising state and event information (col. 3, line 23-40); and an abstraction engine for automatically retrieving and parsing the matrix to generate test cases (e.g. Fig. 3, 4; Fig. 11).

But Okayasu does not disclose generating mapped expected results while parsing the matrix; nor does Okayasu disclose separating test cases based on layers of the software component; nor does Okayasu disclose associating data structures with the test cases, the data structures such that these allow the layer test cases to be uncorrelated. But all these limitations have been addressed in claim 1; and will be referred thereto for the corresponding rejections.

As per claim 41, Okayasu discloses device readable medium embodying a program instructions operable to perform testing of a software component, comprising:

storing (abstraction matrix),

parsing (matrix) as set forth and addressed in claim 1.

But Okayasu does not disclose generating mapped expected results while parsing the matrix; nor does Okayasu disclose separating test cases based on layers of the software component; nor does Okayasu disclose associating data structures with the test cases, the data structures such that these allow the layer test cases to be uncorrelated. Nor does Okayasu teach employing the component in executable form to generate test case execution threads for each layer and executing in parallel some of the execution threads for at least one of the layer, thereby testing the software component.

But all these limitations have been addressed in claim 1; and will be referred thereto for the corresponding rejections.

As per claims 42-50, these claims correspond to the method claims 4-11, and 14, respectively, hence are rejected using the corresponding rejection as set forth therein, respectively.

As per claim 51, this claim is a computer readable medium claim including limitations corresponding to the corresponding limitations (i.e. storing an abstraction matrix, parsing, separating) of system claim 20; hence is rejected using the corresponding rejection as set forth therein.

As per claim 52-54, these claims correspond to the method claims 8, 11, and 14, respectively, hence are rejected using the corresponding rejection as set forth therein, respectively.

Conclusion

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please consult Examiner before using this number)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
August 5, 2004

KAKALI CHAK
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER